CS 4100/5100 – Foundations of Artificial Intelligence

Assignment 2: Procedural Content Generation with Answer Set Programming

Due: October 11th, 5:59pm

Learning Objectives

- Further mastery of knowledge representation in logic programming
- Understanding of declarative programming with constraints
- Introduce concepts of procedural content generation

Resources

- We will be using the answer set solver Clingo, part of the Potassco tool set. It is available on Windows CCIS lab machines in C:\Clingo
- If you would like to run Clingo on your own machine, you can download it from here: http://sourceforge.net/projects/potassco/files/clingo/3.0.4/
    - Note that some people have difficulty running it under Mac OS X

- The instruction manual for Clingo (including sample code) is here: http://www.cs.utexas.edu/~vl/teaching/lbai/clingo_guide.pdf
- Adam Smith's tutorial on answer set programming for procedural content generation is here: http://eis-blog.ucsc.edu/2011/10/map-generation-speedrun/
- An excellent compilation of answer set programs to solve common (and not so common…) problems is here: http://www.hakank.org/answer_set_programming/

Assignment

In this assignment, you will be exploring the use of answer set programming to create content for games. By content, I mean any kind of aspect of a game that a player interacts with: weapons, characters, puzzles, levels, and even rules. Answer set programming is a useful technique for procedural content generation because it allows for the declarative representation of the *space* of potential content that can be created, and adding and removing rules makes that space highly controllable.

You will be modifying the map generation code that we covered in class, and that is available in the tutorial linked above, to add some new features to it. Note that you must also modify the script I provided (or write your own, in a language of your choice, as long as it can run on the machines in WVH 102) to make sure the output of your program is viewable in map format.

You are not required to use the starter code.

Grading Criteria

Choose some features from the list below (or make up your own instead – post on piazza to get permission for a feature you think of) to add to your generator. Undergraduate students are required to choose a total of 3 of these features; graduate students are required to choose 4.

Note that each feature has parameters associated with it that must be set as constants at the top of your file, to make it easy to tweak them. Variables are noted in italics.

- *Some* volcanoes, with surrounding lava with a variable *radius*, that makes a radius around the volcano impassable (i.e. not solid).
- A number of bases equal to the *number of players* of our hypothetical game, which should be a certain *distance* apart from each other and should be reachable.
    - If you are including the volcano feature, there is a *minimum distance* that all bases must be from all volcanoes, and bases are not allowed to be placed on top of lava.
- A variable *number* of mines scattered throughout the map. Mines are not passable.
    - If you are including the forests feature, there may not be any forest within a certain *radius* of the mines.
- A variable *number* of forests. Forests are resources that can be harvested by the player; each forest tile provides a variable *amount* of wood. Forests should be placed such that they can guarantee a *desired quantity* of wood for the players. Forests are passable.
    - If you are including the bases feature, there must be at least one forest tile within a certain *radius* of all bases.
    - 2 bonus percentage points: If you are including the river feature, maximize the number of forest tiles that are near the river.
- A single river that runs through the map. For the sake of simplicity, assume that rivers are passable.
    - If you are including the volcano feature, the river cannot be within a certain *distance* of the lava.
    - If you are including the mines feature, mines cannot be placed within a certain *distance* of the river.
    - 2 bonus percentage points: make rivers impassable, and instead state that a river must have at least two bridges across it that are passable.

Partial credit can only be given if you submit well-commented code that describes the feature you are intending to implement and how you are intending to perform it.

Hints

- Be careful with the numbers you assign to variables, make sure that it is a map that can actually exist in real life. The hardest part of ASP is debugging, since the solver cannot tell you *why* there is no solution, only that there is not one.
- Be careful with how big your map is – too small, and it won't be big enough for all the features. Too large, and it may take too long to solve.
- Add features one at a time, and once you are sure it works, add the next one with the other feature commented out (unless there are dependencies) to be sure that everything works independently before putting it together.
- The "asciify" script is designed to work by piping output from clingo to it, so that you can view the output in a way that makes it easier to understand what is being generated. Make sure that you are #show-ing all relevant predicates, or they will not appear in your visualized output!

Submission Instructions

Submit a zip file containing:

- Your commented program (you can use whatever file extension you'd like) containing your map generator.
- A short write-up describing the features you implemented, any requirements for running your program beyond the starter code, a key for understanding your map (e.g. V corresponds to Volcanoes), and a sample of three noticeably different map layouts that your program can create (in the ASCII representation, not the ASP representation). If you included any features in addition to the ones listed in the assignment, list what they are and why you chose them.
- A readme file with your name, how many late days (if any) you wish to use, and a list of any people you received help from.

All materials must be submitted through Blackboard. **Assignments emailed to me will not be accepted.**